UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/748,098	12/21/2000	Nicholas J. Kelsey	20880-05093; Case 5093	4360

758 7590 04/13/2006

FENWICK & WEST LLP
SILICON VALLEY CENTER
801 CALIFORNIA STREET
MOUNTAIN VIEW, CA 94041

EXAMINER	
HUISMAN, DAVID J	
ART UNIT	**PAPER NUMBER**
2183	

DATE MAILED: 04/13/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/748,098	KELSEY ET AL.	
	Examiner	Art Unit	
	David J. Huisman	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 February 2006*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-55* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-55* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *21 December 2000* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *19 September 2006*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ :
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-55 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: IDS as received on 9/19/2005 and Amendment as received on 2/2/2006.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1, 29-32, and 42-46 are rejected under 35 U.S.C. 103(a) as being unpatentable over Joy et al, U.S. Patent Number 6,542,991 (as applied in the previous Office Action and herein referred to as Joy) in view of Jones et al., U.S. Patent No. 6,317,774 (herein referred to as Jones).

5. Referring to claim 1, Joy has taught a computer based system for switching between program contexts comprising:

a) a processor (Joy figure 3, column 8 lines 14-67) capable of having a first program thread and a second program thread in an execution pipeline having a thread selection hardware (Joy figure 6, column 13 lines 5-23, column 15 lines 4-7);

b) a first set of data storage devices capable of storing a first thread state of said processor (Joy figure 5 number 510 column 13 lines 5-23);

c) a second set of data storage devices capable of storing a second thread state of said processor (Joy figure 5 number 512 column 13 lines 5-23); and

d) while Joy has taught hardware thread scheduler for identifying which of said program threads said processor executes (Joy figure 6 column 15 lines 4-7) and configurable to allocate available processing time of the processor among at least the first and second threads, and a number of varying scheduling concepts (column 3, lines 33-51), Joy has not taught thread-switching at a fixed time according to a predetermined fixed schedule, said schedule specifying that the first thread should be allocated processing time every first number of cycles and that the second thread should be allocated processing time every second number of cycles, wherein said first number of cycles is not equal to said second number of cycles. However, Jones has taught such a concept. Specifically, Jones has taught precomputing a thread schedule and then repeating that schedule. See the abstract. Furthermore, Jones has taught that the threads may execute at different frequencies. For instance, see column 6, line 44, to column 7, line 44. From Table 1 and the passage it can be seen that the second activity/thread, A_B, is allocated 2 units of processing time every 10 cycles. Similarly, thread A_E is allocated 6 units of processing time every 30 cycles. Consequently, it can be seen that different threads are allocated different amounts of processing time. Jones has taught that since his scheduler utilizes a precomputed schedule that specifies the future execution of activities and threads having outstanding time constraints, it is able to overcome the shortcomings of conventional schedulers by significantly reduce the processing required to (A) identify the next thread to execute when the processor

becomes available and (B) determine the amount of time for which to execute the identified thread. As a result, the process of identifying the next thread to execute and determining the amount of time for which to execute the identified thread can be performed in a bounded amount of time that is independent of the number of threads and activities being scheduled. See column 2, lines 19-34. As a result, in order to achieve these benefits, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Joy to include such a scheduler. One would have been motivated to do so because conventional schedulers are inferior to the one set forth by Jones. Also, Joy already states (in column 3, lines 45-50) that a thread-switch decision may be based on a thread timer signal, which would be useful in implementing Jones, since Jones switches threads based on timing.

6. Referring to claim 29, Joy in view of Jones has taught a system as described in claim 1, wherein said thread selection hardware in the embedded pipelined processor switches between said first and second thread state after the end of the execution of a first program instruction in the first thread and before the beginning of the execution of a second program instruction. This is deemed inherent because thread A will execute for some amount of time and then a switch will occur to another thread. The switching marks the end of executing an instruction from thread A and the beginning of executing an instruction in thread B.

7. Referring to claim 30, Joy in view of Jones a system as described in claim 1, wherein said processor is an embedded pipelined processor. See Joy, column 7, lines 52-54.

8. Referring to claim 31, Joy in view of Jones has taught a system as described in claim 1, wherein said first state is the state of the processor during the execution of the first program thread (Joy column 3 line 66-column 4 line 35).

9. Referring to claim 32 Joy in view of Jones has taught a system as described in claim 1, wherein said second state is the state of the processor during the execution of the second program thread (Joy column 3 line 66-column 4 line 35).

10. Referring to claim 42 Joy in view of Jones has taught a system as described in claim 1, wherein said processor is capable of restoring said second state of said processor during execution of said first program thread (Joy column 6 lines 15-35).

11. Referring to claim 43 Joy in view of Jones has taught a system as described in claim 1, wherein said processor is capable of storing said second state of said processor during execution of said first program thread (Joy column 3 lines 3-10; the information of state 2 will still be stored during the execution of state 1).

12. Referring to claim 44 Joy in view of Jones has taught a system as described in claim 1, wherein said first set of data storage devices comprises registers shared by a plurality of threads (Joy column 3 lines 3-10).

13. Referring to claim 45 Joy in view of Jones has taught a system as described in claim 1, wherein the fixed schedule is one of a fixed strict schedule, a semi-flexible strict schedule, and a loose strict schedule. Jones can be considered any one of the above. It is strict because the threads follow some form of rules as to when they may execute.

14. Referring to claim 46, Joy has taught a computer based method for switching between program contexts in a multithreading pipelined processor (Joy figure 3, column 8 lines 14-67) having a hardware thread selector and an execution pipeline, the method comprising:

a) storing a first context of said processor in a first set of data storage devices comprising a first thread state corresponding to a first program thread (Joy figure 5 number 510 column 13 lines 5-23);

b) storing a second context of said processor in a second set of data storage devices comprising a second thread state corresponding to a second program thread (Joy figure 5 number 512 column 13 lines 5-23);

c) while Joy has taught switching the processor from the first thread state to the second thread state by coupling the execution pipeline from the first set of data storage devices to the second set of storage devices via the hardware thread selector at a fixed time according to a predetermined fixed execution schedule (column 3, lines 33-51), Joy has not taught thread-switching at a fixed time according to a predetermined fixed schedule, said schedule specifying that the first thread should be allocated processing time every first number of cycles and that the second thread should be allocated processing time every second number of cycles, wherein said first number of cycles is not equal to said second number of cycles. However, Jones has taught such a concept. Specifically, Jones has taught precomputing a thread schedule and then repeating that schedule. See the abstract. Furthermore, Jones has taught that the threads may execute at different frequencies. For instance, see column 6, line 44, to column 7, line 44. From Table 1 and the passage it can be seen that the second activity/thread, A_B, is allocated 2 units of processing time every 10 cycles. Similarly, thread A_E is allocated 6 units of processing time every 30 cycles. Consequently, it can be seen that different threads are allocated different amounts of processing time. Jones has taught that since his scheduler utilizes a precomputed schedule that specifies the future execution of activities and threads having outstanding time

constraints, it is able to overcome the shortcomings of conventional schedulers by significantly reduce the processing required to (A) identify the next thread to execute when the processor becomes available and (B) determine the amount of time for which to execute the identified thread. As a result, the process of identifying the next thread to execute and determining the amount of time for which to execute the identified thread can be performed in a bounded amount of time that is independent of the number of threads and activities being scheduled. See column 2, lines 19-34. As a result, in order to achieve these benefits, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Joy to include such a scheduler. One would have been motivated to do so because conventional schedulers are inferior to the one set forth by Jones. Also, Joy already states (in column 3, lines 45-50) that a thread-switch decision may be based on a thread timer signal, which would be useful in implementing Jones, since Jones switches threads based on timing.

15. Claims 2-3, 13-17, 19, and 21-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Joy, as applied above, in view of Emer et al., U.S. Patent No. 6,493,741 (as applied in the previous Office Action and herein referred to as Emer).

16. Referring to claim 17 Joy has taught a computer based system for switching between program contexts comprising:

A pipelined processor (Joy figure 3, column 8 lines 14-67) capable of having a first program thread and a second program thread in an execution pipeline having a thread selection hardware (Joy figure 6, column 13 lines 5-23, column 15 lines 4-7);

A first set of data storage devices capable of storing a first thread state of said processor (Joy figure 5 number 510 column 13 lines 5-23);

A second set of data storage devices capable of storing a second thread state of said processor (Joy figure 5 number 512 column 13 lines 5-23); and

A hardware thread scheduler for identifying which of said program threads said processor executes (Joy figure 6 column 15 lines 4-7) and configurable to allocate available processing time of the pipelined processor among at least the first and second states according to a fixed schedule. See column 3, lines 33-51, and note that Joy's thread-switching may be of the oblivious type, in which threads are switched every N cycles without notification of stalling. Since the threads are switched every N cycles, they are switched at a fixed time.

Joy has not explicitly taught that said thread selection hardware in the pipelined processor switches between said first and second thread state between consecutive instruction cycles in response tot the hardware thread scheduler identifying which of said program threads said processor executes. However, recall that Joy has taught that threads may be switched every N cycles. Since there is no disclosed restriction as to what value N might be, a thread switch may occur every cycle (N=1). When N=1, a fine-grained multithreaded system is achieved, as is known in the art. Emer has taught a fine-grained system in which threads are switched every cycle (in consecutive cycles). See Fig.1(b) and column 1, lines 52-65. Such a system eliminates vertical waste, thereby increasing throughput. Furthermore, in some instances, stalls for a particular thread are at least partially, and sometimes fully, masked. That is, if a thread may stall for a maximum period of X cycles, then if the system includes X threads, the stall will never affect the system. For example, assume that a system has 4 threads, and a thread may stall for a

maximum of 4 cycles. In cycle 0, thread 0 must stall, in cycle 1, thread 1 will execute, in cycle 2, thread 2 will execute, in cycle 3, thread 3 will execute, and then in cycle 4, by the time thread 0 is to execute again, the stall time would have elapsed. The stall is effectively masked due to the switching every cycle. As a result, in order to obtain the above advantages, and since Joy discloses that threads may be switched every N cycles, it would have been obvious to one of ordinary skill in the art at the time of the invention to have N=1 and have Joy switch threads on consecutive cycles, as taught by Emer.

17. Referring to claim 2 Joy has taught wherein said first state is the state of the processor during the execution of the first program thread (Joy column 3 line 66-column 4 line 35).

18. Referring to claims 3 Joy has taught wherein said second state is the state of the processor during the execution of the second program thread (Joy column 3 line 66-column 4 line 35).

19. Referring to claims 13 Joy has taught wherein said processor is capable of restoring said second state of said processor during execution of said first program thread (Joy column 6 lines 15-35).

20. Referring to claims 14 Joy has taught wherein said processor is capable of storing said second state of said processor during execution of said first program thread (Joy column 3 lines 3-10; the information of state 2 will still be stored during the execution of state 1).

21. Referring to claims 15 Joy has taught wherein said first set of data storage devices comprises registers shared by a plurality of threads (Joy column 3 lines 3-10).

22. Referring to claims 16 Joy has taught wherein the fixed schedule is one of a fixed strict schedule, a semi-flexible strict schedule, and a loose strict schedule. From column 3, lines 28-

51, switching every N cycles is considered a fixed strict schedule, i.e., a thread switch must occur every N cycles.

23. Referring to claim 19 Joy has taught a computer based method for switching between program contexts in a multithreading pipelined processor (Joy figure 3, column 8 lines 14-67) having a hardware thread selector and an execution pipeline, the method comprising:

Storing a first context of said processor in a first set of data storage devices, the first context corresponding to a first program thread (Joy figure 5 number 510 column 13 lines 5-23);

Storing a second context of said processor in a second set of data storage devices, the second context corresponding to a second program thread (Joy figure 5 number 512 column 13 lines 5-23);

Joy has not explicitly taught switching the processor from executing the first program thread to executing the second program thread between the end of an execution cycle and before the beginning of a next consecutive execution cycle by coupling the execution pipeline from the first set of data storage devices to the second set of storage devices via the hardware thread selector. However, recall that Joy has taught that threads may be switched every N cycles (column 3, lines 33-38). Since there is no disclosed restriction as to what value N might be, a thread switch may occur every cycle (N=1). When N=1, a fine-grained multithreaded system is achieved, as is known in the art. Emer has taught a fine-grained system in which threads are switched every cycle (in consecutive cycles). See Fig.1(b) and column 1, lines 52-65. Such a system eliminates vertical waste, thereby increasing throughput. Furthermore, in some instances, stalls for a particular thread are at least partially, and sometimes fully, masked. That is, if a thread may stall for a maximum period of X cycles, then if the system includes X threads, the

stall will never affect the system. For example, assume that a system has 4 threads, and a thread may stall for a maximum of 4 cycles. In cycle 0, thread 0 must stall, in cycle 1, thread 1 will execute, in cycle 2, thread 2 will execute, in cycle 3, thread 3 will execute, and then in cycle 4, by the time thread 0 is to execute again, the stall time would have elapsed. The stall is effectively masked due to the switching every cycle. As a result, in order to obtain the above advantages, and since Joy discloses that threads may be switched every N cycles, it would have been obvious to one of ordinary skill in the art at the time of the invention to have N=1 and have Joy switch threads on consecutive cycles, as taught by Emer.

24. Referring to claim 21 Joy has taught further comprising:

Identifying which of the said program threads said processor executes according to an execution schedule (Joy figure 6 column 15 lines 4-7).

25. Referring to claim 22 Joy has taught further comprising allocating available processing time of the processor among at least the first and second threads according to the execution schedule (Joy figure 6, column 2 lines 40-45).

26. Referring to claim 23 Joy has taught wherein the allocating comprises dividing the available execution time into a plurality of quanta, each quanta corresponding to a number of instruction cycles for execution of a thread (Joy figure 6, column 3, lines 33-36; each thread executes for N cycles).

27. Referring to claim 24 Joy has taught wherein at least one quanta corresponds to a thread that is scheduled to execute periodically after a fixed number of execution cycles (Joy figure 6, column 3, lines 33-36).

28. Claims 4 and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Joy in view of Emer, as applied above, and further in view of Borkenhagen et al, U.S. Patent Number 6,567,839 (as applied in the previous Office Action and herein referred to as Borkenhagen).

29. Referring to claims 4 and 20, Joy in view of Emer has not taught wherein said processor switches between said first and second states by changing a state selection register. Borkenhagen has taught wherein said processor switches between said first and second states by changing a state selection register (Borkenhagen column 16 lines 38-49).

It would have been obvious to one of ordinary skill in the art at the time of the invention to use a register to change the threads of a multithreaded system. The system of Joy must have some mechanism to control the thread switch, but does not disclose details of this logic. Borkenhagen does disclose a particular embodiment for this logic and states reasoning to use this type of switching logic for such reason as assigning certain threads priority, which ties into the Joy system which has hard real time threads (Borkenhagen abstract, column 5 line 66-column 6 line 11). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to look to Borkenhagen to use a register to change the threads of a multithreaded system if they want to do a detailed design since Joy does not describe that logic function.

30. Claims 5-9, 18, and 25-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Joy in view of Emer, as applied above, and further in view of Ramakrishnan et al, U.S. Patent Number 6,085,215 (as applied in the previous Office Action and herein referred to as Ramakrishnan).

31. Referring to claims 5, and 25-26, Joy in view of Emer has not explicitly taught wherein said hardware thread scheduler includes:

A thread identifier for identifying at least one hard-real-time (HRT) thread and at least one non-real-time thread;

A HRT scheduler for regularly scheduling said HRT thread in available time quanta such that said HRT thread is scheduled to ensure the execution of the HRT in a predetermined time. However, Ramakrishnan has taught wherein said thread scheduler includes:

A thread identifier for identifying at least one hard-real-time (HRT) thread and at least one non-real-time thread (Ramakrishnan figures 2, 2a, and 2b, abstract, column 5 line 54-column 6 line 8, column 9 lines 9-21, column 15 line 39-column 16 line 6, column 8 lines 47-53);

A HRT scheduler for regularly scheduling said HRT thread in available time quanta such that said HRT thread is scheduled to ensure the execution of the HRT in a predetermined time (Ramakrishnan figures 2, 2a, and 2b, abstract, column 5 line 54-column 6 line 8, column 9 lines 9-21, column 15 line 39-column 16 line 6, column 8 lines 47-53).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have a thread scheduler that handles real time threads so that real time threads will be executed when needed for critical systems needing real time operations. Ramakrishnan has taught this need for scheduling and has taught a solution with real time thread scheduling with general purpose threads and real time threads (Ramakrishnan column 1 lines 10-32, column 3 line 25-column 4 line 6, column 5 line 54-column 6 line 8). This will allow systems requiring real time operations to execute the required functions when needed, and the real time threads will have priority. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the

invention to have a thread scheduler that handles real time threads to allow for real time threads to be completed when required.

32. Referring to claim 6 the combination of Joy, Emer, and Ramakrishnan has taught wherein said time quanta is at least one instruction cycle (Ramakrishnan figures 2, 2a, and 2b, abstract, column 5 line 54-column 6 line 8, column 9 lines 9-21, column 15 line 39-column 16 line 6, column 8 lines 47-53; any switching would require some time, so it would take at least one unit of time, or one cycle).

33. Referring to claims 7 and 27 the combination of Joy, Emer, and Ramakrishnan has taught wherein said thread scheduler schedules a non-real-time (NRT) thread to replace a scheduled HRT thread if said HRT is complete (Ramakrishnan column 5 line 54-column 6 line 8, column 9 lines 9-21).

34. Referring to claim 8 the combination of Joy, Emer, and Ramakrishnan has taught wherein said thread scheduler schedules the execution of non-real-time (NRT) threads in quanta not allocated to HRT threads (Ramakrishnan column 5 line 54-column 6 line 8, column 9 lines 9-21).

35. Referring to claims 9 and 28 the combination of Joy, Emer, and Ramakrishnan wherein said thread scheduler regularly schedules NRT threads to be executed (Ramakrishnan column 5 line 54-column 6 line 8, column 9 lines 9-21).

36. Referring to claim 18 the combination of Joy, Emer, and Ramakrishnan has taught wherein said time quanta is exactly one instruction cycle (Ramakrishnan figures 2, 2a, and 2b, abstract, column 5 line 54-column 6 line 8, column 9 lines 9-21, column 15 line 39-column 16

line 6, column 8 lines 47-53; column 9 discusses how one time slot is used for the general purpose domain). Also, Emer shows a time quanta of one cycle for each thread (Fig.1(b)).

37. Claims 10-12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Joy in view Emer in view of Ramakrishnan, as applied above, and further in view of Gutgold et al., U.S. Patent Number 6,026,503 (as applied in the previous Office Action and herein referred to as Gutgold).

38. Referring to claim 10 the combination of Joy, Emer, and Ramakrishnan has not taught a first storage device for storing program instructions, said processor fetching instructions from the first storage device within a first fetch period;

A second storage device for storing program instructions, said processor fetching instructions from the second storage device within a second fetch period;

Wherein said first fetch period is substantially shorter than said second fetch period. Gutgold has taught wherein a first storage device for storing program instructions, said processor fetching instructions from the first storage device within a first fetch period;

A second storage device for storing program instructions, said processor fetching instructions from the second storage device within a second fetch period;

Wherein said first fetch period is substantially shorter than said second fetch period (Gutgold column 3 lines 1-19; it is well known in the art that flash memory is not as fast in operation as RAM memory, therefore having a slower fetch period than that of the RAM memory fetches).

It would have been obvious to one of ordinary skill in the art at the time of the invention to use two different memories to fetch instructions from. Gutgold has taught storing the debug executable information in the EEPROM, or ROM, and that having a debug mode for a processor is beneficial since it allows the user to test the code they have written just as programmers test code on an emulator (Gutgold column 1 lines 27-45 and column 2 lines 20-26). By being able to test the code, the user can make sure the code runs efficiently and correctly. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to be able to debug the program code just as an emulator would, allows the program to be tested to make sure that is correct and does not causes errors.

39. Referring to claim 11 the combination of Joy, Emer, and Ramakrishnan and Gutgold has taught wherein said first storage device for storing program instructions comprises a static RAM (Gutgold column 3 lines 1-19).

40. Referring to claim 12 the combination of Joy, Emer, and Ramakrishnan and Gutgold has taught wherein said second storage device for storing program instructions comprises a flash memory (Gutgold column 3 lines 1-19).

41. Claims 33 and 47 are rejected under 35 U.S.C. 103(a) as being unpatentable over Joy in view of Jones, as applied above, and further in view of Borkenhagen, as applied above.

42. Referring to claim 33 and 47 Joy in view of Jones has not taught wherein said processor switches between said first and second states by changing a state selection register. Borkenhagen has taught wherein said processor switches between said first and second states by changing a state selection register (Borkenhagen column 16 lines 38-49).

It would have been obvious to one of ordinary skill in the art at the time of the invention to use a register to change the threads of a multithreaded system. The system of Joy must have some mechanism to control the thread switch, but does not disclose details of this logic. Borkenhagen does disclose a particular embodiment for this logic and states reasoning to use this type of switching logic for such reason as assigning certain threads priority, which ties into the Joy system which has hard real time threads (Borkenhagen abstract, column 5 line 66-column 6 line 11). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to look to Borkenhagen to use a register to change the threads of a multithreaded system if they want to do a detailed design since Joy does not describe that logic function.

43. Claims 34-38 and 48-55 are rejected under 35 U.S.C. 103(a) as being unpatentable over Joy in view of Jones, as applied above, in view of Ramakrishnan, as applied above.

44. Referring to claim 34 Joy in view of Jones has not explicitly taught wherein said hardware thread scheduler includes:

A thread identifier for identifying at least one hard-real-time (HRT) thread and at least one non-real-time thread;

A HRT scheduler for regularly scheduling said HRT thread in available time quanta such that said HRT thread is scheduled to ensure the execution of the HRT in a predetermined time. However, Ramakrishnan has taught wherein said thread scheduler includes:

A thread identifier for identifying at least one hard-real-time (HRT) thread and at least one non-real-time thread (Ramakrishnan figures 2, 2a, and 2b, abstract, column 5 line 54-column 6 line 8, column 9 lines 9-21, column 15 line 39-column 16 line 6, column 8 lines 47-53);

A HRT scheduler for regularly scheduling said HRT thread in available time quanta such that said HRT thread is scheduled to ensure the execution of the HRT in a predetermined time (Ramakrishnan figures 2, 2a, and 2b, abstract, column 5 line 54-column 6 line 8, column 9 lines 9-21, column 15 line 39-column 16 line 6, column 8 lines 47-53).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have a thread scheduler that handles real time threads so that real time threads will be executed when needed for critical systems needing real time operations. Ramakrishnan has taught this need for scheduling and has taught a solution with real time thread scheduling with general purpose threads and real time threads (Ramakrishnan column 1 lines 10-32, column 3 line 25-column 4 line 6, column 5 line 54-column 6 line 8). This will allow systems requiring real time operations to execute the required functions when needed, and the real time threads will have priority. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to have a thread scheduler that handles real time threads to allow for real time threads to be completed when required.

45. Referring to claim 35 the combination of Joy and Jones and Ramakrishnan has taught wherein said time quanta is at least one instruction cycle (Ramakrishnan figures 2, 2a, and 2b, abstract, column 5 line 54-column 6 line 8, column 9 lines 9-21, column 15 line 39-column 16 line 6, column 8 lines 47-53; any switching would require some time, so it would take at least one unit of time, or one cycle).

46. Referring to claim 36, the combination of Joy and Jones and Ramakrishnan has taught wherein said thread scheduler schedules a non-real-time (NRT) thread to replace a scheduled

HRT thread if said HRT is complete (Ramakrishnan column 5 line 54-column 6 line 8, column 9 lines 9-21).

47. Referring to claim 37 the combination of Joy and Jones and Ramakrishnan has taught wherein said thread scheduler schedules the execution of non-real-time (NRT) threads in quanta not allocated to HRT threads (Ramakrishnan column 5 line 54-column 6 line 8, column 9 lines 9-21).

48. Referring to claim 38 the combination of Joy and Jones and Ramakrishnan wherein said thread scheduler regularly schedules NRT threads to be executed (Ramakrishnan column 5 line 54-column 6 line 8, column 9 lines 9-21).

49. Referring to claims 48 and 52-53 Joy in view of Jones has not explicitly taught wherein said hardware thread scheduler includes:

A thread identifier for identifying at least one hard-real-time (HRT) thread and at least one non-real-time thread;

A HRT scheduler for regularly scheduling said HRT thread in available time quanta such that said HRT thread is scheduled to ensure the execution of the HRT in a predetermined time. However, Ramakrishnan has taught wherein said thread scheduler includes:

A thread identifier for identifying at least one hard-real-time (HRT) thread and at least one non-real-time thread (Ramakrishnan figures 2, 2a, and 2b, abstract, column 5 line 54-column 6 line 8, column 9 lines 9-21, column 15 line 39-column 16 line 6, column 8 lines 47-53);

A HRT scheduler for regularly scheduling said HRT thread in available time quanta such that said HRT thread is scheduled to ensure the execution of the HRT in a predetermined time

(Ramakrishnan figures 2, 2a, and 2b, abstract, column 5 line 54-column 6 line 8, column 9 lines 9-21, column 15 line 39-column 16 line 6, column 8 lines 47-53).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have a thread scheduler that handles real time threads so that real time threads will be executed when needed for critical systems needing real time operations. Ramakrishnan has taught this need for scheduling and has taught a solution with real time thread scheduling with general purpose threads and real time threads (Ramakrishnan column 1 lines 10-32, column 3 line 25-column 4 line 6, column 5 line 54-column 6 line 8). This will allow systems requiring real time operations to execute the required functions when needed, and the real time threads will have priority. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to have a thread scheduler that handles real time threads to allow for real time threads to be completed when required.

50. Referring to claim 49, Joy and Jones has taught allocating available processing time of the processor among at least the first and second threads according to the predetermined fixed execution schedule. See Joy column 3, lines 33-36.

51. Referring to claim 50, Joy and Jones has taught wherein the allocating comprises dividing the available execution time into a plurality of quanta, each quanta corresponding to a number of instruction cycles for execution of a thread. See Joy column 3, lines 33-36.

52. Referring to claim 51, Joy and Jones has taught wherein at least one quanta corresponds to a thread that is scheduled to execute periodically after a fixed number of execution cycles. See column 3, lines 33-36.

53. Referring to claim 54 the combination of Joy and Jones and Ramakrishnan has taught wherein said thread scheduler schedules a non-real-time (NRT) thread to replace a scheduled HRT thread if said HRT is complete (Ramakrishnan column 5 line 54-column 6 line 8, column 9 lines 9-21).

54. Referring to claim 55 the combination of Joy and Jones and Ramakrishnan wherein said thread scheduler regularly schedules NRT threads to be executed (Ramakrishnan column 5 line 54-column 6 line 8, column 9 lines 9-21).

55. Claims 39-41 are rejected under 35 U.S.C. 103(a) as being unpatentable over Joy in view of Jones in view of Ramakrishnan, as applied above, and further in view of Gutgold, as applied above.

56. Referring to claim 39 the combination of Joy and Jones and Ramakrishnan has not taught a first storage device for storing program instructions, said processor fetching instructions from the first storage device within a first fetch period;

A second storage device for storing program instructions, said processor fetching instructions from the second storage device within a second fetch period;

Wherein said first fetch period is substantially shorter than said second fetch period. Gutgold has taught wherein a first storage device for storing program instructions, said processor fetching instructions from the first storage device within a first fetch period;

A second storage device for storing program instructions, said processor fetching instructions from the second storage device within a second fetch period;

Wherein said first fetch period is substantially shorter than said second fetch period (Gutgold column 3 lines 1-19; it is well known in the art that flash memory is not as fast in operation as RAM memory, therefore having a slower fetch period than that of the RAM memory fetches).

It would have been obvious to one of ordinary skill in the art at the time of the invention to use two different memories to fetch instructions from. Gutgold has taught storing the debug executable information in the EEPROM, or ROM, and that having a debug mode for a processor is beneficial since it allows the user to test the code they have written just as programmers test code on an emulator (Gutgold column 1 lines 27-45 and column 2 lines 20-26). By being able to test the code, the user can make sure the code runs efficiently and correctly. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to be able to debug the program code just as an emulator would, allows the program to be tested to make sure that is correct and does not causes errors.

57. Referring to claim 40 the combination of Joy and Jones and Ramakrishnan and Gutgold has taught wherein said first storage device for storing program instructions comprises a static RAM (Gutgold column 3 lines 1-19).

58. Referring to claim 41 the combination of Joy and Jones and Ramakrishnan and Gutgold has taught wherein said second storage device for storing program instructions comprises a flash memory (Gutgold column 3 lines 1-19).

Response to Arguments

59. Applicant's arguments filed on February 2, 2006, have been fully considered but they are not persuasive.

60. Applicant argues the novelty/rejection of claim 17 on page 18 of the remarks, in substance that:

> "The deficiencies of Joy are not rectified by Emer. Emer discloses a multithreaded architecture in which thread switching occurs every cycle. As discussed above, the frequency of thread switching should not be confused with the overhead pf thread switching. While Emer may disclose switching every execution cycle, Emer does not disclose switching between consecutive instruction cycles. In fact, Emer acknowledges that switching overhead has not been completely eliminated in the disclosed multithreaded architecture: "Multithreaded processors better tolerate long-latency operations, effectively eliminating vertical waste." (col. 1, ln. 58-60, emphasis added). As vertical waste has been only effectively eliminated, Emer suggests that in fact some number of cycles goes completely unused in the process of switching from one thread to another. (See Emer's definition of vertical waste, col. 1, ln. 41-42)"

61. These arguments are not found persuasive for the following reasons:

a) It is not clear where Emer states that some cycles go completely unused in the process of switching. As stated by Emer, vertical waste is eliminated and this is shown in Fig.1(b) as no cycles go unused. The only way vertical waste can exist in this situation is if there less total threads than cycles in a latency event, as described in the rejection (e.g. if there are 5 threads and a latency event requires four cycles, that latency will never affect the machine). However, any number of threads can be implemented in this type of system and applicant has not limited the claims one way or the other. So, vertical waste is eliminated by Emer. In addition, Emer has taught both switching every cycle and between instruction cycles. It should be realized that applicant has not defined an instruction cycle. An instruction cycle may simply be the period of time that the each thread actually executes. The cycle does not have to include switching.

Therefore, in order to finish executing a first thread and begin executing a second thread, switching must occur between those instruction cycles.

Conclusion

62. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Binns, "A Robust High-Performance Time Partitioning Algorithm: The Digital Engine Operating System (DEOS) Approach," 2001, has taught a static time-partitioned system in which threads execute (Fig.1).

Chung et al., U.S. Patent No. 5,404,469, has taught a multi-threaded microprocessor architecture utilizing static interleaving, wherein each function unit is allocated to a particular thread in a fixed, predetermined time slot, in a repeating pattern of time slots.

Applicant is reminded that in amending in response to a rejection of claims, the patentable novelty must be clearly shown in view of the state of the art disclosed by the references cited and the objections made. Applicant must also show how the amendments avoid such references and objections. See 37 CFR § 1.111(c).Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after

the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
April 5, 2006

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100